UNITED STATES PATENT APPLICATION

FOR

# AN ADAPTIVE WEB SERVER

INVENTORS:

Nina T. Bhatti
Tarek F. Abdelzaher

Prepared by:

Thomas X. Li (Reg. No. 37,079)
Hewlett-Packard Company
Corporate Legal Department, 20BN
3000 Hanover Street
Palo Alto, California 94304
(650) 857-5972
thomas_li@hp.com

Attorney's Docket No. 10982229

"Express Mail" mailing label number: **EM198753708US**
Date of deposit: **4/26/99**
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and is addressed to the Assistant Commissioner of Patents, Washington, D.C. 20231.

Terry Flores
(Typed or Printed name of person mailing paper or fee)

Terry Flores
(Signature of person mailing paper or fee)

# AN ADAPTIVE WEB SERVER

A1 ~~BACKGROUND OF THE INVENTION~~

## 1. Field of the Invention

The present invention pertains to Internet/Intranet systems. More particularly, this invention relates to an adaptive web server that serves user access requests for the web server to less resource intensive contents in the web server when the web server is overloaded.

## 2. Description of the Related Art

An example of a data access network system is the Internet or Intranet network system. An Internet/Intranet network system typically includes a number of data service systems connected together via interconnect networks. The data service systems typically include web and/or content servers that host contents for various customers or applications. The customers are the owners of the contents hosted in the data service systems such that subscribers or users can access the contents via their computer terminals. The content servers typically utilize Internet applications, such as electronic mail, bulletin boards, news groups, and World Wide Web access. The hosted contents are arranged in the form of content sites within the content servers. Each site may include a number of pages (e.g., world wide web pages). A content site is typically for one customer of the server while a particular customer may own a number of content sites. The content servers may also be referred to as web servers.

Often, a web page is formed from a number of data and/or executable

program files, such as text files, graphics files, audio files, video files, and application program files. Each of the files is referred to as an "object". The web pages can be accessed by a user at a user terminal (e.g., a personal computer system or a web access device) connected to any one of the data service systems.

As is known, access to the web pages via Internet is typically structured around the HTTP (Hyper Text Transfer Protocol) protocol. The HTTP protocol is a request-and-response protocol. When a user at a client device designates a particular web page, at least one request is generated. The number of requests is dependent upon the sophistication of the designated web page. As described above, a web page may include one or more "objects". A multi-object page is aesthetically pleasing, but each object requires a separate request and a separate response. Therefore, the time for each request-and-response round trip plays a role in determining the total time a user must wait to view the complete web page.

A web server can be accessed by multiple users at the same time. The web server typically handles the user access requests in the first-come-first-served (FIFO) fashion. One disadvantage of this type of prior art web server structure is that the web server is not equipped with protection mechanism against excessive load conditions. As is known, a web server typically has limits on the number of requests it can handle per second. In addition, the web server also has limits on the response time and bandwidth that can be served. Under high loads, the web server can be overwhelmed with requests, resulting in longer response time and poor performance for users. Thus, when the number of user access requests to the web server greatly exceeds the processing

capacity of the web server (i.e., overloaded), the web server either handles a request in an unbearably long period of time or simply becomes non responsive. The sharp increase in server response time may also cause user connections to time out, creating the perception that the server is down or crashed. In mission-critical applications, such as e-commerce, perceived server downtime may result in loss of sales and/or other financial losses.

## SUMMARY OF THE INVENTION

One feature of the present invention is to improve performance of a web server.

Another feature of the present invention is to increase performance of a web server by serving user access requests for the web server from less resource intensive contents in the web server when the web server is to be overloaded.

A further feature of the present invention is to provide an adaptive web server that, when in overload situation, routes the user access requests to the less resource intensive contents in the web server such that the web server does not become overloaded or non-functional.

A data service system in a data access network system includes a content server that stores content files for access by external access requests. Each of the content files is stored in a full content format and any number of adapted or degraded content formats which are less resource-intensive to serve than the full content format. The data service system also includes an adaptive load control system which is coupled to the content server to pass the access requests to the content server. The adaptive load control system modifies an access request to access the corresponding content file in one of the adapted content formats when the content server is in an overload condition such that the content server can be maintained at safe load conditions.

In a data service system of a data access network system having a content server that stores content files for access by external access requests, a method of maintaining the content server at safe load conditions includes the step of determining load condition of the content server when the data service system

receives an access request to access one of the content files stored in the content server. This step can be performed internally in the content server or external to the content server. If the content server is determined to be in an overload condition, then the access request is modified to access the corresponding content file in an adapted content format which is less resource-intensive to serve than the same file in a full content format such that the content server is maintained at the safe load conditions.

Other features and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a data access network system that includes a number of data service systems.

Figure 2 shows the structure of one of the adaptive data service systems of Figure 1 in accordance with one embodiment of the present invention.

Figure 3 shows the control and feedback loop of the adaptive data service system of Figure 2 for partial degradation.

Figure 4 schematically shows the structure of the content server of the adaptive data service system of Figure 2.

Figures 5 and 6 show various comparisons between the adaptive data service system of Figure 2 and a prior art data service system.

Figure 7 shows a Gap Estimation method employed by a load monitor of Figures 2-3.

## DETAILED DESCRIPTION OF THE INVENTION

Figure 1 shows a data access network system 10. In one embodiment, the network system 10 is an Internet system. In another embodiment, the network system 10 is an Intranet system. Alternatively, the data access network system 10 may be any other known network system that employs a known communication protocol.

As can be seen from Figure 1, the data service system 20 is connected to a number of user terminals 11a through 11n via an interconnect network 12. Users at each of the user terminals 11a-11n can access the data service system 20 for the services provided by the data service system 20. The users at each of the user terminals 11a-11n can also access the data service system 15 for the services provided by the data service system 15 via the data service system 20 and the Internet 13. The user terminals 11a-11n can also be referred to as client devices.

Each of the user terminals 11a-11n may be located at a residence, a school, or an office of the user. Each of the user terminals 11a-11n includes a network access application program (e.g., a web browser application program such as Netscape's Navigator) that allows the user to access the data services offered by the data service systems 15 and 20.

Each of the user terminals 11a-11n can be a computer system or other electronic device with data processing capabilities (e.g., a web TV). The interconnect network 12 can be any known network, such as Ethernet, ISDN (Integrated Services Digital Network), T-1 or T-3 link, FDDI (Fiber Distributed Data Network), cable or wireless LMDS network or telephone line network.

Figure 1 shows only two data service systems 15 and 20 for the data access network system 10 for illustration purposes only. In practice, the data access network system 10 may include many more data service systems. In addition, the Internet 13 is formed by a number of data service systems connected together via a network. Data communications among all data service systems are conducted using a predetermined communication protocol for Internet/Intranet communications. In one embodiment, the communication protocol is the Hyper Text Transport Protocol (HTTP). Alternatively, other known communication protocols for Internet/Intranet communications can also be used.

Each of the data service systems 15 and 20 has substantially the same functional structure, which will be described in more detail below. Each of the data service systems 15 and 20 can be employed by an Internet/Intranet Service Provider (ISP) to offer data services (e.g., web, news, or advertisement) and other services (e.g., e-commerce, e-mail) to users or subscribers connected to the data access system 10 via one of the data service systems 15 and 20.

Each of the data service systems 15 and 20 includes a number of servers (e.g., web servers, e-mail servers, news servers, e-commerce servers, domain name servers, address assignment servers, proxy servers, advertisement servers, and session manager servers). The web servers, e-mail servers, news servers, e-commerce servers, and advertisement servers can be collectively referred to as local service servers or content servers. A content server typically stores a number of content files that include Hyper-Text Markup Language (HTML) web pages, gif images, video clips, etc. Data transfers to and from the content servers are enabled by transport protocols such as Transport Control Protocol

(TCP) and the User Datagram Protocol (UDP). The content servers support a variety of Internet applications to provide services such as access to the World Wide Web, electronic mail, bulletin boards, chat rooms, news groups, and e-commerce. Using a currently commercially available web browser and other client applications, users via their respective user terminals can access the content files (e.g., web pages, news, images, e-mails,) stored in the content servers.

As described above, the access to the content servers is structured around the HTTP protocol which is a request-and-response protocol. When a user at a user terminal desires to access a content file stored in a content server of one of the data service systems 15 and 20, at least one request is generated and sent to the content server. The content server can handle multiple requests at the same time. However, the content server has limits on the number of requests it can handle per second. When the number of requests received by the content server greatly exceeds the limits of the content server, the content server will be in an overload condition, resulting in at least unbearably longer response time and poor performance for users. This means when the request rate increases beyond the server capacity, server performance deteriorates dramatically, potentially causing a service outage. In addition, when a client issues a request and does not get any response, the client is likely to issue several more subsequent requests. This quickly increases the number of requests received by the content server. To resolve the problem, each of the data service systems 15 and 20, in accordance with one embodiment of the present invention, includes an adaptive load control mechanism to control the load condition of its content servers such that each content server does not become overloaded. Alternatively, the load

control mechanism of the present invention can also be applied to other servers of each of the data service systems 15 and 20. The adaptive load control mechanism is described in more detail below, also in conjunction with Figures 2-6.

As can be seen from Figure 2, a data service system 30 in accordance with one embodiment of the present invention is shown to include a content server 31. The data service system 30 also includes an adaptive load control system 40 that controls the load condition of the content server 31 in accordance with one embodiment of the present invention. Thus, the data service system 30 can also be referred to as adaptive data service system or adaptive web server. The data service system 30 can be any one of the data service systems (e.g., the data service system 15 or 20) of Figure 1.

As will be described in more detail below, the content server 31 stores content files or dynamic executable code/program for access by access requests. Thus, the content files hereinafter refer to (1) static content files, (2) dynamic content files, and (3) executable programs/codes. Each of the content files is stored in a full content format and an adapted or degraded content format. A content file in the adapted content format is much smaller in size and less resource intensive than the same file in the full content format. The adaptive load control system 40 passes the access requests to the content server 31 when receiving the requests. When the content server 31 is in an overload condition or about to be in the overload condition, the adaptive load control system 40 modifies the access requests to access their corresponding content files in the adapted content format. When the adaptive load control system 40 determines that the content server 31 is not in the overload condition, the adaptive load

control system 40 modifies the access requests to access their corresponding content files in the full content format. This allows the content server 31 to be maintained at safe load conditions. Alternatively, the adaptive load control system 40 can adapt the content server 31 based not on the load condition of the content server 31, but on the client's abilities (e.g., network connection of the client device, latency, or display capability of the client device). The data service system 30 will be described in more detail below, also in connection with Figures 2-6.

The data service system 30 can be implemented in a computer system or other data processing system. The computer system that implements the data service system 30 can be server computer system, a workstation computer system, a personal computer system, or a mainframe computer system, a notebook computer system, or any other computer system.

The data service system 30 includes a network interface 35 that interfaces the adaptive load control system 40. The adaptive load control system 40 then interfaces with the content server 31. The network interface 35 serves as the interface of the data service system 30 with external network (not shown). The interface 35 can be implemented by any known network interface technology, and thus will not be described in more detail below. The network interface 35 receives external requests for the content server 31 and passes the requests to the content server 31 via the adaptive load control system 40.

The content server 31 can be any kind of content server that stores a number of content files. Each of the content files can be accessed by an access request. The content server 31 may also include a number of content sites, each storing a number of content files for access by multiple access requests. The

multiple content sites may belong to different content providers or customers with potential conflicting interests. In this case, the adaptive load control system 40 causes the content server 31 to be a number of performance-isolated virtual content servers, which will be described in more detail below. Each of the servers has a capability to provide independent performance guarantees regardless of the load on the other virtual servers.

The content server 31 can be a static server or dynamic server. In one embodiment, the content server 31 is a static content server that stores static content files only. In another embodiment, the content server 31 may store both static and dynamic content files. As is known, web content is generally classified as static, such as a file, or dynamic, such as cgi-scripts. Dynamic content may be generated at run-time by a back-end engine (e.g., a database engine) that is separate from the server itself. Any adaption at the back-end engine is not handled by the adaptive load control system 40.

In accordance with one embodiment of the present invention, each of the content files stored in the content server 31 is stored in multiple formats or versions. For example, a content file can be stored in its original full content format or version. The same content file is also stored in a degraded or adapted format or version in the content server 31. Figure 4 shows an example of such arrangement. As can be seen from Figure 4, the content file in the degraded format 72 is smaller in size and worse in viewing quality than the same file in the full content format or version 71. This means that the degraded format of the content file requires less resource (e.g., bandwidth) to serve than the counterpart full content format of the same content file. This is possible because a content file can be shown, for example, at 64×64 pixel resolution, at 128×128

pixel resolution, at 256×256 pixel resolution, or at higher or lower resolutions. The 256×256 pixel resolution version of a content file clearly has more image data and requires more resource to transfer than the 128×128 pixel resolution version of the same content file. Thus, if the full content version of a content file is at the 256×256 pixel resolution, the degraded version of the same content file can be at 128×128 pixel resolution, or at 64×64 pixel resolution. In summary, the degraded format of a content file may include images of less resolution, fewer embedded images, simplified or simpler pages without backgrounds or fewer hyper-links than the full content format of the same content file. Alternatively, each of the content files stored in the content server 31 has several degraded versions of the same content file. The content server 31 will be described in more detail below with only two versions (i.e., the full content version and one degraded version), for example, for each of the content files stored in the content server 31 for illustration purposes only.

Referring again to Figure 2, each access request specifies a content file stored in the content server 31 using the Universal Resource Locator (URL). The URL path for a content file stored in the content server 31, however, does not specify which version or format from which the content file will be accessed. This means that multiple content trees contain different versions of the same URL. The path to a particular URL in a given content tree will be the concatenation of the content tree name and the URL name, prefixed by the name of the root service directory of the content server 31. For example, two content trees may be created in the root service directory "/root", one being "/full_content" and the other "/degraded_content". The "/full_content" content tree name directs the request to access the corresponding content file in the full

Sub C1

content version or format. The "/degraded_content" content tree name directs the request to access the corresponding content file in the degraded content version or format. In this case, a URL of "http://www.hpl.hp.com/my_picture.jpg" may be served from either the directory "/root/full_content/my_picture.jpg", or from the directory "root/degraded_content/my_picture.jpg".

The scheme applies even if the content file is dynamic content, such as the content generated by cgi-scripts. Multiple content trees may contain different versions or formats of the named cgi-scripts (e.g., "my_script.cgi"). The "/degraded_content/" version or format is a less resource-intensive search script that, for example, looks only for the first five matches, instead of the "/full_content" version or format that looks for one hundred matches. The URL of the script is then prefixed by the right tree name to determine which version of the script to execute under given load conditions. For example, the URL of "http://www.hpl.hp.com/my_script.cgi" may be modified to become either "http://www.hpl.hp.com/root/full_content/cgi-bin/my_script.cgi" or "http://www.hpl.hp.com/root/degraded_content/cgi-bi/my_script.cgi".

Alternatively, the less resource-intensive "/degraded_content" search script can be replaced or substituted with a static version or format. This can be done by switching to a different content tree. For example, an on-line vendor using the content server 31 can use a dynamically generated version of their product catalog that interacts with a stock database to display the items currently in stock. When the content server 31 is overloaded, its statically pre-stored catalog version is then accessed and displayed. The reason for doing this is that

a dynamic script, even degraded or simplified, still consumes more resources to serve than the static content.

The adaptive load control system 40 is connected to the content server 31. In accordance with one embodiment of the present invention, the adaptive load control system 40 detects overload condition of the content server 31 and causes the access requests to access the corresponding file from one of several alternative content qualities for delivery in accordance with the load conditions. When the adaptive load control system 40 detects that the content server 31 is in the overload conditions, the adaptive load control system 40 causes the incoming access requests to the content server 31 to access the corresponding content files in the degraded format which is less resource-intensive to serve. The degraded format reduces the size of responses. The degraded format also reduces the total number of requests as the simplified content includes fewer embedded images and/or other objects. The degraded format also removes or reduces links embedded in the requested content file. Thus, the degraded format is less resource intensive to serve than the full content format.

The adaptive load control system 40 is also able to reject access requests if needed. In addition, the adaptive load control system 40 supports co-existence of multiple performance-isolated virtual servers such that performance levels and load adaption decisions of virtual server do not affect that of another virtual server. This is required when the content server 31 hosts multiple content sites that possibly belong to different customers or content providers with potentially conflicting interests. The total load on the server 31 may not be evenly distributed among the hosted sites. In this case, the adaptive load control system 40 implements a performance isolation mechanism that does not penalize

a site for excessive load incurred by another site. Using this mechanism, the content server 31 is viewed as a collection of one or more performance-isolated virtual servers, each with a capability to provide independent performance guarantees regardless of the load on other virtual servers, if any. Guarantees are declared in a service level agreement associated with each virtual server and are used to configure the adaptive load control system 40 for appropriate resource capacity allocation. The load adaptive control system 40 makes adaption decisions within each virtual server of the content server 31 on its load and resource allocation.

Moreover, the adaptive load control system 40 also gives preferential treatment to guaranteed service classes, while allowing non-guaranteed service classes to consume the excess capability when available. The guaranteed service classes are governed by service level agreements. The non-guaranteed classes are typically served on best effort basis with available resources. Under overload conditions, the adaptive load control system 40 should direct non-guaranteed requests to their degraded content files first. An arbitrary policy can govern the relative degradation of client classes with different importance levels. For example, under overload conditions, the class A requests should be sent to the degraded content files before class B requests. But if resources remain scarce, class B requests are then degraded before rejecting class A requests.

The adaptive load control system 40 can be implemented at any point in the data service system 30 where the server's requests and responses can be accessed. This means that the adaptive load control system 40 can be implemented in the web server software, in the UNIX socket library, or in the operating system of a computer system that embodies the data service system

30. Moreover, the adaptive load control system 40 can also be implemented in the gateway that is external to the server, as well as transparent to the server.

Basically, the adaptive load control system 40 requires three entry points. They are (1) initialization point, (2) request pre-processing point, and (3) request post-processing point. The initialization point initializes internally used data structures, and creates a separate process that implements various components of the adaptive load control system 40 if desired. The request pre-processing point decides the quality of content level of the given request. The content adaptor 41 of the adaptive load control system 40 is called from this point. The content adaptor 41 may communicate with the other components via shared memory. The request post-processing point is an optional point which monitors response size.

If server code is not available, the adaptive load control system 40 can be implemented in a middleware (e.g., a dynamically linked socket library) used by the server 31. In a UNIX environment, as part of server initialization, the server socket is created and a listen() call is made to the socket library to listen for incoming requests. This library routine can be modified to call the initialization point of the middleware. The modification can be transparent to the content server 31. The functional structure of the adaptive load control system 40 is described in more detail below, also in conjunction with Figures 2-7.

As can be seen from Figure 2, the adaptive load control system 40 includes a request classifier 33 connected to the network interface 35. The adaptive load control system 40 also includes a load monitor 32 connected to the request classifier 33. A content adapter 41 is then connected to the load monitor 32 and the content server 31. An adaption controller 50 is connected to the

content adapter 41 and the load monitor 32. The components 32-33, 41, and 50 together form the adaptive load control system 40. Alternatively, the adaptive load control system 40 may include more or fewer components than above mentioned. For example, the adaptive load control system 40 may function without the request classifier 33.

The request classifier 33 receives the access requests from the network interface 35. The request classifier 33 is used to classify the received access requests into a number of classes based on different classification metrics. The request classifier 33 allows preferential treatment to a subset of content sites or clients. The request classifier 33 is used in an installation where quality of service differentiation is required among different classes of clients (e.g., guaranteed and non-guaranteed clients, or clients accessing different virtual servers). In this situation, the action taken by the content adapter 41 later may depend on the identity of the client or the requested content. The request classifier 33 allows multiple classes of users to share the same content file or site and yet receive different treatments or performance. Class-based service is a mechanism for differentiating services given to individual classes. Thus, service performance can be priced based on performance or service agreements. A higher class with greater guarantee can be priced higher than a lower class that may offer less guarantee and more "best effort" services. The request classifier 33 can be implemented using any known technology.

The request classifier 33 then sends the classified access requests to the load monitor 32. The load monitor 32 is used to monitor the load condition of the content server 31 in order to detect overload condition. The load monitor 32 employs a number of monitoring mechanisms to monitor the load condition of

the content server 31. One way of monitoring the load condition of the content server 31 is to monitor the response time of the content server 31. This is because response time is proportional to the length of the input request queue of the content server 31 (e.g., the server's socket listen queue in a UNIX implementation). When the server is under-loaded, the queue tends to be short (or empty), resulting in short response time. When the content server 31 is overloaded, the request queue within the content server 31 overflows, making the response time grow an order of magnitude. This serves as a clear overload indicator. The advantage of this approach (i.e., estimating queue length by monitoring response time) is that the mechanism can be implemented outside the server process and requires no modification to the code of the content server 31.

Another way of monitoring whether the content server 31 is overloaded is to monitor utilization of the server machine, as well as the response time. This allows the load monitor 32 to determine how under-utilized the server is. When utilization decreases below a predetermined level, load monitor 32 may determine the content server 31 is no longer in the overload condition.

In this case, in order to provide a faithful server load estimate, utilization measurement should be indicative of the load attributed to useful work (i.e., serving client requests), rather than the aggregated utilization. For example, the existence of a low priority process or thread that implements a busy-waiting loop on some event will render the utilization identically 100% even in the absence of load on the server. Another concern arises when all server threads or processes have been committed to current requests and are blocked in I/O. The resulting idle time cannot be taken advantage of since all resources are already utilized and therefore should not be measured towards available capacity.

To account for the above mentioned concerns, two different mechanisms are developed for the load monitor 32. The first is referred to as the Linear Approximation Method. Using this method, the load monitor 32 employs a linear function of measured request rate R and delivered bandwidth BW to determine the system utilization U, consumed by processing client requests. The linear function is shown as follows.

$$U = aR + bBW$$

wherein constants a and b can be computed by either on-line or off-line priori profiling as will be described below. The function is good for estimating offered load as long as it does not exceed server capacity. When measuring in the overload condition, the linear approximation method is invalid. Thus, the linear approximation method is combined with the response time monitoring to determine the actual load condition of the server 31. This combined load monitoring function means that if the measured response time is above the predetermined overload threshold, U = 100%. If the measured response time is below the predetermined overload threshold, U = aR + bBW.

One advantage of this linear approximation method is that it is easy to implement. Using this method, each server process or thread Tj records independently its observed request rate Rj, delivered bandwidth BWj, and utilization Uj. A separate monitor process or thread then reads and sums up the recorded values to compute the aggregate request rate R, bandwidth BW, and utilization U. Synchronization of access to shared data structures is not required since there is only one writer to any piece of data. The method allows server capacity planning for quality of service guarantees, and it provides means for converting the desired request rate and bandwidth guarantee into a

corresponding resource capacity allocation (i.e., allocated utilization). This method, however, requires computing the constants a and b using the priori profiling.

A simple way of computing the constants a and b using the priori profiling is to obtain several measurements of U and the corresponding R and BW. Then a linear regression technique is employed to determine the constants a and b that best fit the equation U =aR + bBW. Note that R and BW can be measured online by counting requests seen and bytes delivered within a given period. Utilization U can be measured using a Gap Estimation method which will be described later. Given the measured values of R, BW, and U, at successive time intervals, estimation theory provides a way to find a linear fit that minimizes the error. It provides the necessary formulae for computing and updating the parameters a and b online in view of successive new measurements.

Alternatively, the parameters a and b can be determined by testing the content server 31 with a pre-specified workload. Testing can proceed by requesting a URL of a given size at an increasing rate until client connections start timing out, indicating server overload. The maximum attained request rate and bandwidth are then recorded. For our purposes, server utilization can be assumed to be 100% at this load condition. The experiment is repeated for different sizes of the requested URL, giving a different rate and bandwidth combination that saturates the server 31. The set of R, BW, and U = 100% points are used to construct the line aR + bBW = 100 on an R, BW, plane. The line intersects the R and BW axes at 100/a and 100/b respectively, from which a and b are found.

The second mechanism for the load monitor 32 is referred to as the Gap

Estimation Method. This method estimates the fraction of time that the content server 31 spends serving requests. Using this method, the load monitor 32 employs a global counter to track the requests. The load monitor 32 increments the counter when a request is received and decrements the counter when a response departs the content server 31. The counter will return to its initial value only when a gap is present (i.e., when all current requests have been served and no additional requests have arrived yet). By summing up the gaps over a period of time T, a monitor process or thread can compute the total "idle time" G within that time period T. The idle time is the time where no requests are pending. The utilization is then estimated as $U = (T-G)/T$.

The Gap Estimation method is illustrated in Figure 7. As can be seen from Figure 7, concurrent processing of request bursts are separated with idle time. The method does not require a prior profiling, and does not require monitoring response time. The global counter may be updated by multiple writers and thus requires some form of locking or access synchronization.

Referring back to Figure 2, the load monitor 32 sends the load condition information of the content server 31 to the content adapter 41 and the adaption controller 50. The adaption controller 50 also receives a predetermined desired load value which indicates the threshold of the overload condition. Based on the comparison of the desired load value and the monitored load value, the adaptive controller 50 determines whether the content server 31 is in the overload condition in which case the adaptive controller 50 causes the content adapter 41 to modify the URLs of all the incoming access requests to access their corresponding content files in the degraded content format or version. If the adaptive controller 50 determines that the content server 31 is not in the

overload condition in which case the adaptive controller 50 causes the content adapter 41 to modify the URLs of the incoming access requests to access their corresponding content files in the full content format or version.

The desired load value for the adaption controller 50 can be a predetermined threshold T of the server response time. This threshold T can be set equal to (or slightly smaller than) the maximum server response time specified in the service level agreement, if any, thereby causing adaption when the agreement is about to be violated. In the absence of such specification, the threshold can be derived from the pre-configured maximum input request queue length Q and the average service time S. For example, if we consider a 90% full queue to be an overload indication, the threshold T can be set to $T = 0.9Q \times S$.

The content adapter 41 effects adaption by changing directory links. Under the control of the adaptive controller 50, the content adapter 41 modifies the URLs of the incoming access requests to access the corresponding content files in either the full content format or one of the degraded content formats. For example, the content adapter 41 can modify the URL of "http://www.hpl.hp.com/my_picture.jpg" as either "http://www.hpl.hp.com/full_content/my_picture.jpg" to access the content file "my_picture.jpg" in the full content format, or "http://www.hpl.hp.com/degraded_content/my_picture.jpg" to access the content file "my_picture.jpg" in the degraded content format. As another example, when the content is a dynamic content (e.g., http://www.hpl.hp.com/my_script.cgi), the content adapter 41 can modify the URL as either "http://www.hpl.hp.com/full_content/cgi-bin/my_script.cgi" to access the full script file, or "http://www.hpl.hp.com/degraded_content/cgi-

bin/my_script.cgi" to access the less resource-intensive script.

When the adaption controller 50 determines that the content server 31 is in the overload condition, the adaption controller 50 causes the content adapter 41 to modify the URLs of the incoming access requests to access the corresponding content files in the degraded or adapted content format. As described above, the degraded or adapted content format requires less resources to serve than the full content format. When the adaption controller 50 determines that the content server 31 is not in the overload condition, the adaption controller 50 causes the content adapter 41 to modify the URLs of the incoming access requests to access the corresponding content files in the full content format.

As can be seen from Figures 2 and 4, once the adaption trigger is fired from the adaption controller 50, the content adapter 41 switches transparently between the high quality service tree 73 (Figure 4) and the degraded quality service tree 74 (Figure 4) in the root service directory 70 (Figure 4) based on the load condition of the content server 31. The switching operation is completely transparent to the content server 31.

Figure 5 shows the result of the adaption process by the adaptive load control system 40. The curve 81 shows the percentage of connection failures by a prior art server without the adaptive load control system 40. The curve 80 shows the percentage of connection failures by the content server 31 with the adaptive load control system 40. As can be seen from Figure 5, the curve 80 shows significantly fewer connection failures than the prior art curve 81 as the request rate increases. The traditional server suffers an increasing error rate (i.e., connection failures) when offered load exceeds capacity of the server.

As described above, when the adaption controller 50 determines that the content server 31 is overloaded, the controller 50 causes the content adapter 41 to adapt all incoming access requests to the degraded content format. This potentially makes the content server 31 under-loaded, thus not achieving adequate resource utilization when the adaption takes place. This effect is shown in Figure 6. In Figure 6, the solid line curve 85 represents the delivered bandwidth of a prior art content server without the adaption mechanism in accordance with one embodiment of the present invention and the broken line curve 86 represents the delivered bandwidth of the content server 31 with the adaptive load control system 40. As can be seen from Figure 6, a sharp drop in delivered bandwidth for the curve 86 is shown for the content server 31 when adaption takes place by switching to less resource-intensive content format. This leaves server bandwidth underutilized.

This can be overcome by only causing the content adapter 41 to degrade a fraction of the incoming access requests when the content server 31 is in the overload condition. The fraction can be between nothing and the all of the incoming access requests. This mechanism is implemented in the content adapter 41. It uses an input control parameter G which can be thought of as an external control knob that tunes the extent of partial degradation. The extremes are G = Max, where no request is degraded, and G = 0, where all incoming requests are degraded. Thus, decreasing G will decrease server load.

Assuming the total number of available content trees are Max and the content trees are numbered from 1 to Max in increasing order of quality (tree 0 is a special tree that stands for request rejection). Let I be the integral part of G and let F be the fractional part. The following rule is used:

If G is an integer (i.e., G=I, F=0), the content adapter 41 lets every request be served from the tree I.

If G is not an integer, the content adapter 41 computes a pseudo-random number N (in the range [0,1]) upon the receipt of each request. If N<F, the request is served from tree I+1. Otherwise it is served from tree I.

This algorithm provides a continuous partial degradation spectrum that ranges between serving all requests from the highest quality content tree and rejecting all requests. Next, a self-regulating technique will be described, in conjunction with Figure 3, that uses monitoring and feedback to automatically select the best value of G such that the overload is prevented and a specified target utilization is maintained.

Figure 3 shows the control and feedback loop of the adaptive load control system 40 of Figure 2 for self-regulating partial degradation in accordance with another embodiment of the present invention. In this embodiment as shown in Figure 3, a partial degradation module 60 is used to control the partial degradation. The module 60 includes a integral controller 60 and an adder 61. The achieved server utilization from the load monitor 32 is fed to the adder 61 of the module 60. The desired load utilization is also fed to the adder 61. The end result E (i.e., the difference between the two utilizations) from the adder 61 is then fed to the integral controller 62. The integral controller 62 then adjusts the control parameter G to control the content adapter 41 to adjust the number of access requests for adaption. Using the feedback, the content adapter 41 can quickly and automatically adjust the number of degraded requests to be around the target or desired utilization.

In the foregoing specification, the invention has been described with

reference to specific embodiments thereof. It will, however, be evident to those skilled in the art that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.